Task_5_prodigy_infotech_internship

June 4, 2024

PRODIGY INFOTECH DATA SCIENCE INTERN

#TASK 5

TASK OVERVIEW: Analyze traffic accident data to identify patterns related to road conditions, weather, and time of day. Visualize accident hotspots and contributing factors.

[ ]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

[ ]:
```
df = pd.read_csv("/content/traffic accident in india.csv")
```

DATA PREPROCESSING

[ ]:
```
df.head()
```

[ ]:
```
       Time Day_of_week Age_band_of_driver Sex_of_driver   Educational_level  \
0  17:02:00      Monday              18-30          Male   Above high school
1  17:02:00      Monday              31-50          Male  Junior high school
2  17:02:00      Monday              18-30          Male  Junior high school
3  01:06:00      Sunday              18-30          Male  Junior high school
4  01:06:00      Sunday              18-30          Male  Junior high school

  Vehicle_driver_relation Driving_experience      Type_of_vehicle  \
0                Employee             1-2yr           Automobile
1                Employee        Above 10yr  Public (> 45 seats)
2                Employee             1-2yr      Lorry (41?100Q)
3                Employee            5-10yr  Public (> 45 seats)
4                Employee             2-5yr                  NaN

  Owner_of_vehicle Service_year_of_vehicle  … Vehicle_movement  \
0            Owner              Above 10yr  …   Going straight
1            Owner                 5-10yrs  …   Going straight
2            Owner                     NaN  …   Going straight
3     Governmental                     NaN  …   Going straight
4            Owner                 5-10yrs  …   Going straight
```

```
    Casualty_class Sex_of_casualty Age_band_of_casualty Casualty_severity  \
0               na              na                   na                na
1               na              na                   na                na
2  Driver or rider            Male                31-50                 3
3       Pedestrian          Female                18-30                 3
4               na              na                   na                na

  Work_of_casuality Fitness_of_casuality Pedestrian_movement  \
0               NaN                  NaN    Not a Pedestrian
1               NaN                  NaN    Not a Pedestrian
2            Driver                  NaN    Not a Pedestrian
3            Driver               Normal    Not a Pedestrian
4               NaN                  NaN    Not a Pedestrian

            Cause_of_accident Accident_severity
0             Moving Backward     Slight Injury
1                  Overtaking     Slight Injury
2   Changing lane to the left    Serious Injury
3  Changing lane to the right     Slight Injury
4                  Overtaking     Slight Injury

[5 rows x 32 columns]
```

[ ]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12316 entries, 0 to 12315
Data columns (total 32 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   Time                         12316 non-null  object
 1   Day_of_week                  12316 non-null  object
 2   Age_band_of_driver           12316 non-null  object
 3   Sex_of_driver                12316 non-null  object
 4   Educational_level            11575 non-null  object
 5   Vehicle_driver_relation      11737 non-null  object
 6   Driving_experience           11487 non-null  object
 7   Type_of_vehicle              11366 non-null  object
 8   Owner_of_vehicle             11834 non-null  object
 9   Service_year_of_vehicle      8388 non-null   object
 10  Defect_of_vehicle            7889 non-null   object
 11  Area_accident_occured        12077 non-null  object
 12  Lanes_or_Medians             11931 non-null  object
 13  Road_allignment              12174 non-null  object
 14  Types_of_Junction            11429 non-null  object
 15  Road_surface_type            12144 non-null  object
 16  Road_surface_conditions      12316 non-null  object
```

```
 17  Light_conditions             12316 non-null  object
 18  Weather_conditions           12316 non-null  object
 19  Type_of_collision            12161 non-null  object
 20  Number_of_vehicles_involved  12316 non-null  int64
 21  Number_of_casualties         12316 non-null  int64
 22  Vehicle_movement             12008 non-null  object
 23  Casualty_class               12316 non-null  object
 24  Sex_of_casualty              12316 non-null  object
 25  Age_band_of_casualty         12316 non-null  object
 26  Casualty_severity            12316 non-null  object
 27  Work_of_casuality            9118 non-null   object
 28  Fitness_of_casuality         9681 non-null   object
 29  Pedestrian_movement          12316 non-null  object
 30  Cause_of_accident            12316 non-null  object
 31  Accident_severity            12316 non-null  object
dtypes: int64(2), object(30)
memory usage: 3.0+ MB
```

[ ]:
```
df.describe()
```

[ ]:
```
       Number_of_vehicles_involved  Number_of_casualties
count                 12316.000000          12316.000000
mean                      2.040679              1.548149
std                       0.688790              1.007179
min                       1.000000              1.000000
25%                       2.000000              1.000000
50%                       2.000000              1.000000
75%                       2.000000              2.000000
max                       7.000000              8.000000
```

[ ]:
```
df.columns
```

[ ]:
```
Index(['Time', 'Day_of_week', 'Age_band_of_driver', 'Sex_of_driver',
       'Educational_level', 'Vehicle_driver_relation', 'Driving_experience',
       'Type_of_vehicle', 'Owner_of_vehicle', 'Service_year_of_vehicle',
       'Defect_of_vehicle', 'Area_accident_occured', 'Lanes_or_Medians',
       'Road_allignment', 'Types_of_Junction', 'Road_surface_type',
       'Road_surface_conditions', 'Light_conditions', 'Weather_conditions',
       'Type_of_collision', 'Number_of_vehicles_involved',
       'Number_of_casualties', 'Vehicle_movement', 'Casualty_class',
       'Sex_of_casualty', 'Age_band_of_casualty', 'Casualty_severity',
       'Work_of_casuality', 'Fitness_of_casuality', 'Pedestrian_movement',
       'Cause_of_accident', 'Accident_severity'],
      dtype='object')
```

[ ]:
```
df.nunique()
```

[ ]:
```
Time                          1074
Day_of_week                      7
Age_band_of_driver               5
Sex_of_driver                    3
Educational_level                7
Vehicle_driver_relation          4
Driving_experience               7
Type_of_vehicle                 17
Owner_of_vehicle                 4
Service_year_of_vehicle          6
Defect_of_vehicle                3
Area_accident_occured           14
Lanes_or_Medians                 7
Road_allignment                  9
Types_of_Junction                8
Road_surface_type                5
Road_surface_conditions          4
Light_conditions                 4
Weather_conditions               9
Type_of_collision               10
Number_of_vehicles_involved      6
Number_of_casualties             8
Vehicle_movement                13
Casualty_class                   4
Sex_of_casualty                  3
Age_band_of_casualty             6
Casualty_severity                4
Work_of_casuality                7
Fitness_of_casuality             5
Pedestrian_movement              9
Cause_of_accident               20
Accident_severity                3
dtype: int64
```

[ ]:
```
df.isnull().sum()
```

[ ]:
```
Time                             0
Day_of_week                      0
Age_band_of_driver               0
Sex_of_driver                    0
Educational_level              741
Vehicle_driver_relation        579
Driving_experience             829
Type_of_vehicle                950
Owner_of_vehicle               482
Service_year_of_vehicle       3928
Defect_of_vehicle             4427
```

```
Area_accident_occured          239
Lanes_or_Medians               385
Road_allignment                142
Types_of_Junction              887
Road_surface_type              172
Road_surface_conditions          0
Light_conditions                 0
Weather_conditions               0
Type_of_collision              155
Number_of_vehicles_involved      0
Number_of_casualties             0
Vehicle_movement               308
Casualty_class                   0
Sex_of_casualty                  0
Age_band_of_casualty             0
Casualty_severity                0
Work_of_casuality             3198
Fitness_of_casuality          2635
Pedestrian_movement              0
Cause_of_accident                0
Accident_severity                0
dtype: int64
```

[ ]:
```
# fill missing values with mean column values
df['Driving_experience'].fillna(df['Driving_experience'].mode()[0],␣
 ↪inplace=True)
df['Age_band_of_driver'].fillna(df['Age_band_of_driver'].mode()[0],␣
 ↪inplace=True)
df['Type_of_vehicle'].fillna(df['Type_of_vehicle'].mode()[0], inplace=True)
df['Area_accident_occured'].fillna(df['Area_accident_occured'].mode()[0],␣
 ↪inplace=True)
df['Road_allignment'].fillna(df['Road_allignment'].mode()[0], inplace=True)
df['Type_of_collision'].fillna(df['Type_of_collision'].mode()[0], inplace=True)
df['Vehicle_movement'].fillna(df['Vehicle_movement'].mode()[0], inplace=True)
df['Lanes_or_Medians'].fillna(df['Lanes_or_Medians'].mode()[0], inplace=True)
df['Types_of_Junction'].fillna(df['Types_of_Junction'].mode()[0], inplace=True)
```

[ ]:
```
df.isnull().sum()
```

[ ]:
```
Age_band_of_driver             0
Driving_experience             0
Type_of_vehicle                0
Area_accident_occured          0
Lanes_or_Medians               0
Road_allignment                0
Types_of_Junction              0
Road_surface_conditions        0
```

```
Light_conditions               0
Weather_conditions             0
Type_of_collision              0
Number_of_vehicles_involved    0
Number_of_casualties           0
Vehicle_movement               0
Casualty_class                 0
Age_band_of_casualty           0
Pedestrian_movement            0
Cause_of_accident              0
Accident_severity              0
TimeInSeconds                  0
dtype: int64
```

[ ]:
```
df.duplicated()
```

[ ]:
```
0        False
1        False
2        False
3        False
4        False
         …
12311    False
12312    False
12313    False
12314    False
12315    False
Length: 12316, dtype: bool
```

[ ]:
```
df.dtypes.value_counts()
```

[ ]:
```
object    30
int64      2
Name: count, dtype: int64
```

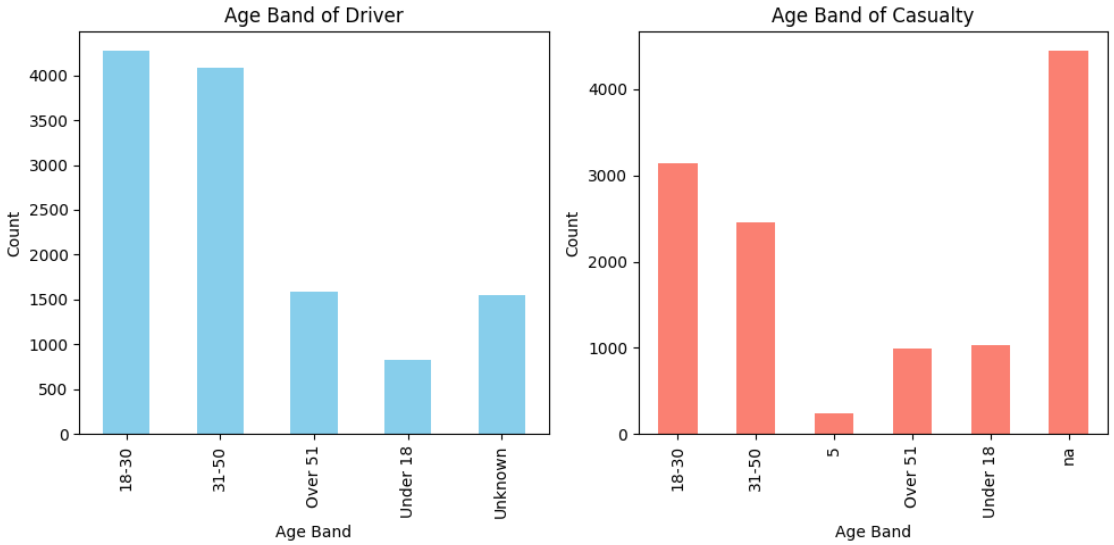
[ ]:
```
# convert the 'Date' column to datetime format
df['Time']= pd.to_datetime(df['Time'])
```

```
<ipython-input-16-e69021690a5a>:2: UserWarning: Could not infer format, so each
element will be parsed individually, falling back to `dateutil`. To ensure
parsing is consistent and as-expected, please specify a format.
  df['Time']= pd.to_datetime(df['Time'])
```

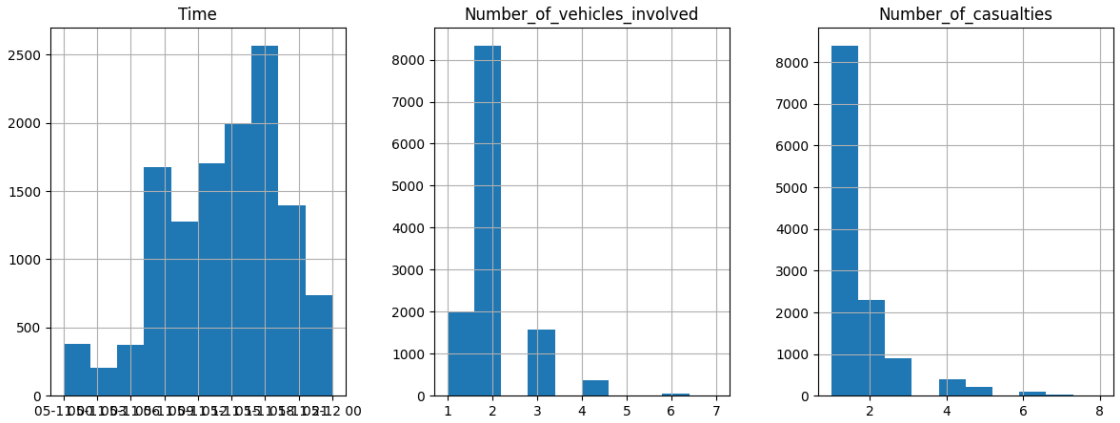
EDA

[ ]:
```
plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
```

```
df['Age_band_of_driver'].value_counts().sort_index().plot(kind='bar',␣
 ↪color='skyblue')
plt.title('Age Band of Driver')
plt.xlabel('Age Band')
plt.ylabel('Count')

#  'Age_band_of_casualty'
plt.subplot(1, 2, 2)
df['Age_band_of_casualty'].value_counts().sort_index().plot(kind='bar',␣
 ↪color='salmon')
plt.title('Age Band of Casualty')
plt.xlabel('Age Band')
plt.ylabel('Count')

plt.tight_layout()
plt.show()
```



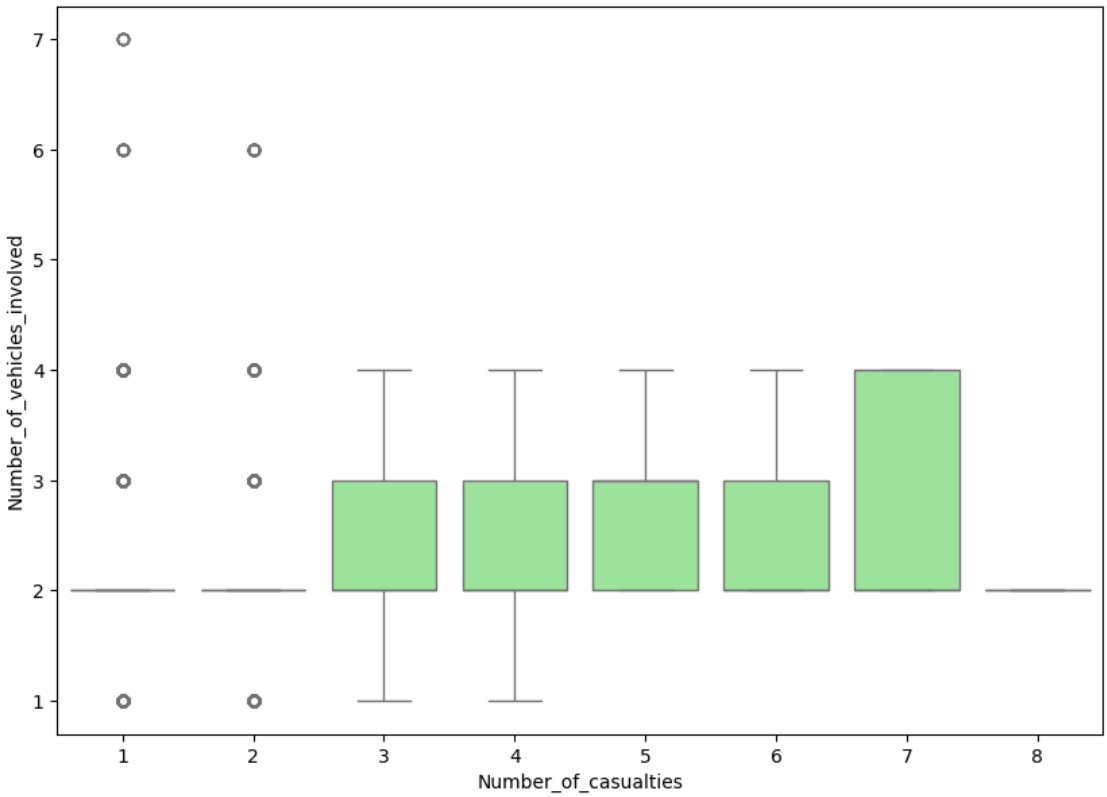
[ ]:
```
df.hist(layout=(1,6), figsize=(30,5))
plt.show()
```



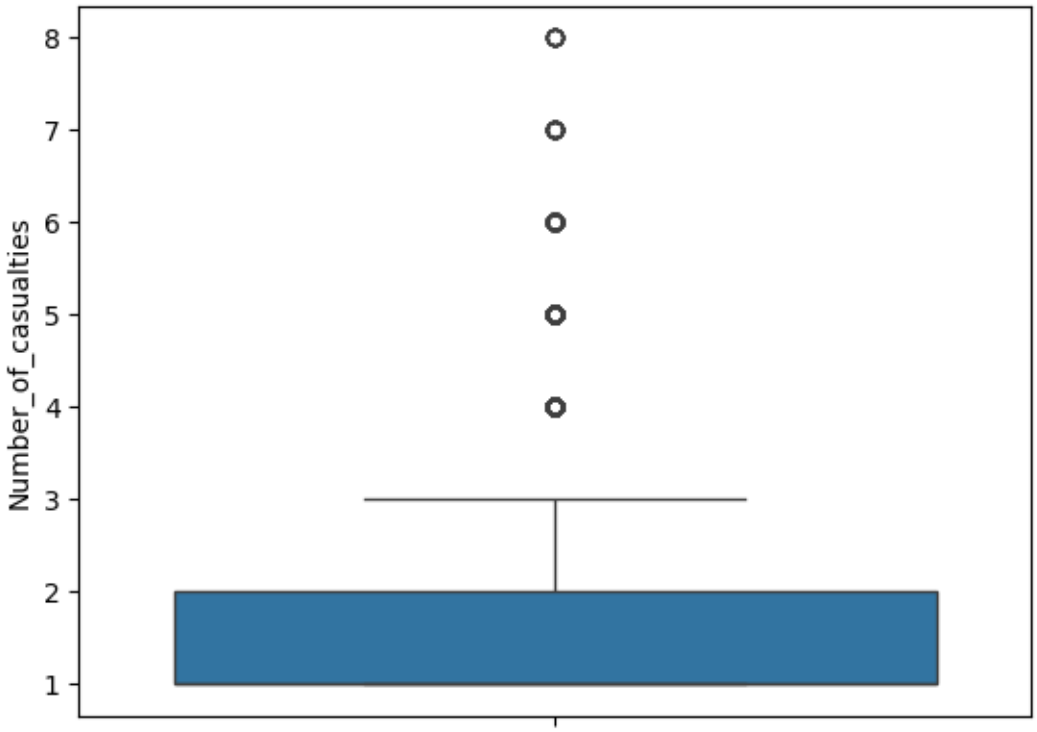
[ ]:
```
df['Number_of_casualties'].value_counts()
```

[ ]:
```
Number_of_casualties
1    8397
2    2290
3     909
4     394
5     207
6      89
7      22
8       8
Name: count, dtype: int64
```

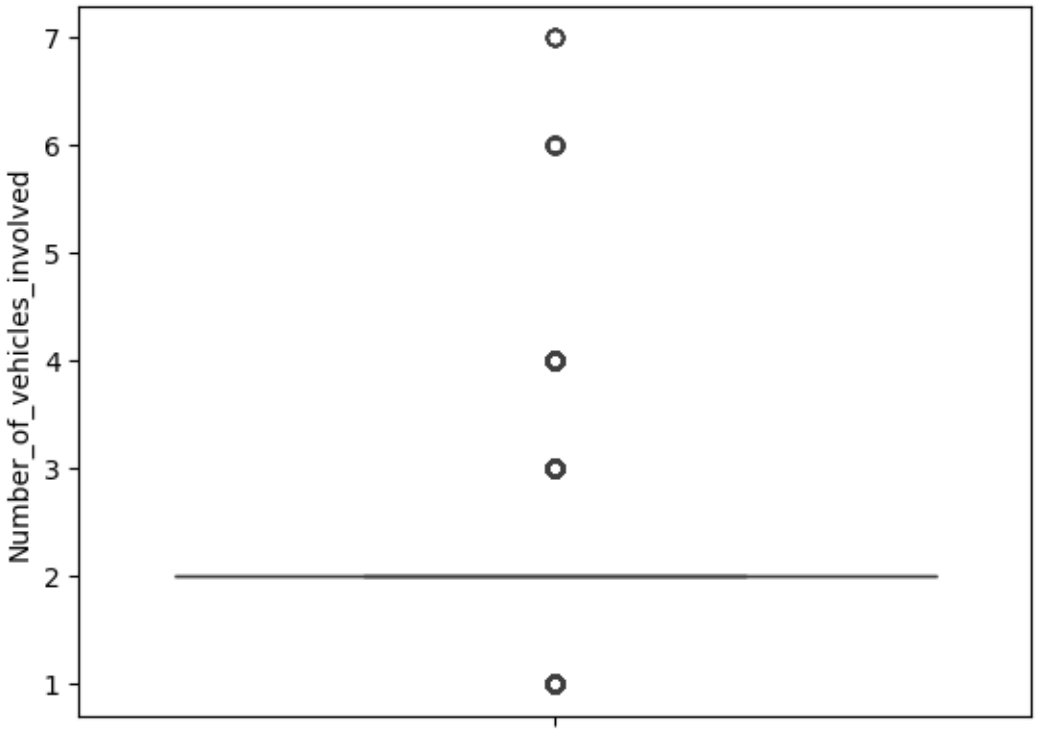
[ ]:
```
plt.figure(figsize=(10,7))
sns.boxplot(data=df, y='Number_of_vehicles_involved',␣
 ↪x='Number_of_casualties',color="lightgreen")
plt.show()
```



[ ]:
```
sns.boxplot(data=df, y='Number_of_casualties')
plt.show()
```



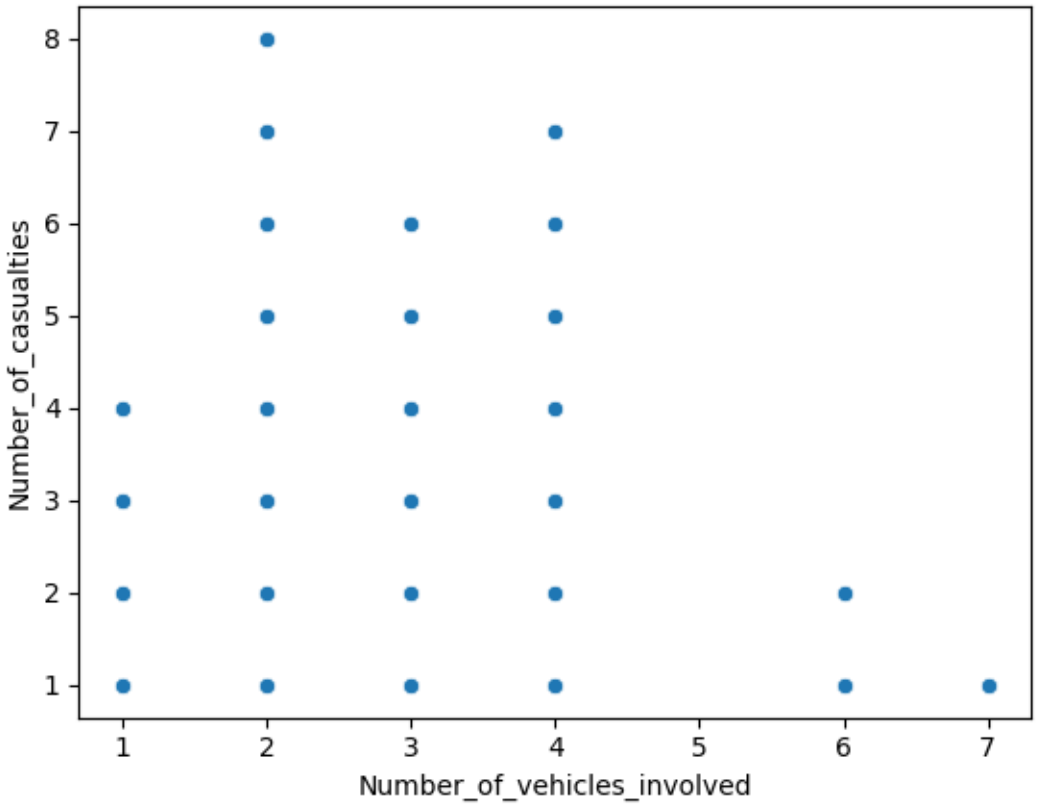
[ ]:
```
sns.boxplot(data=df, y='Number_of_vehicles_involved')
plt.show()
```



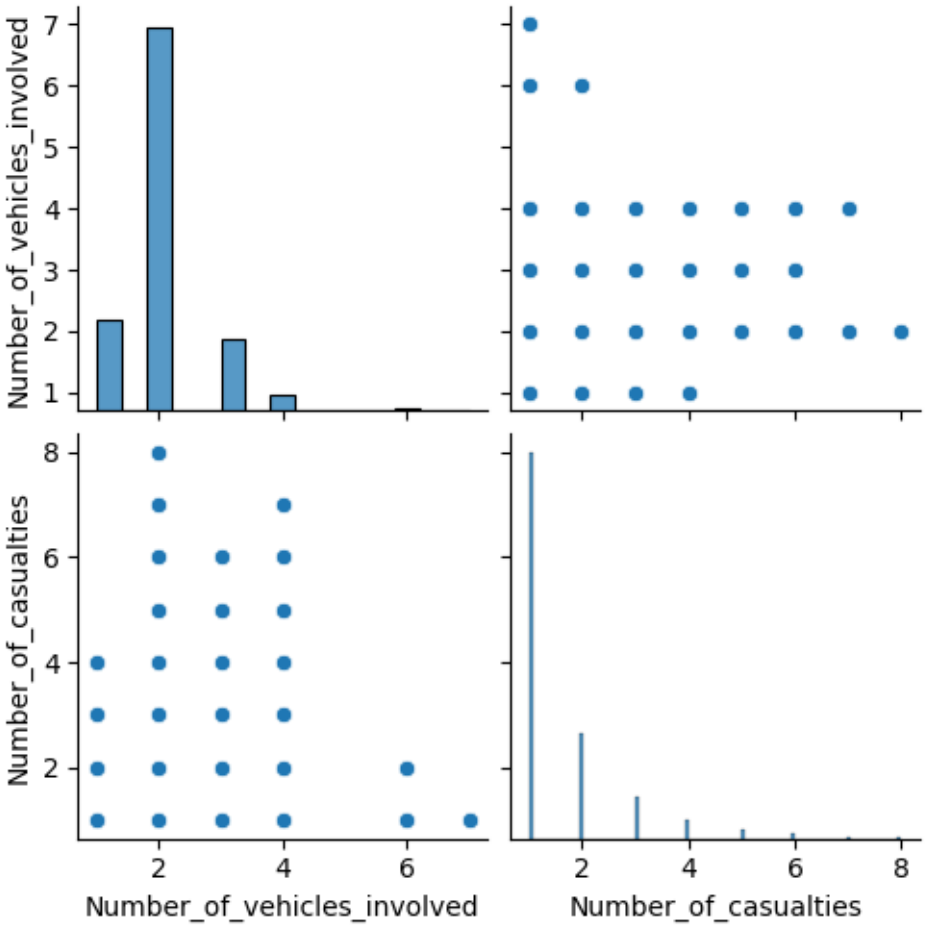
[ ]:
```
df['Number_of_vehicles_involved']
```

[ ]:
```
0        2
1        2
2        2
3        2
4        2
        ..
12311    2
12312    2
12313    1
12314    2
12315    2
Name: Number_of_vehicles_involved, Length: 12316, dtype: int64
```

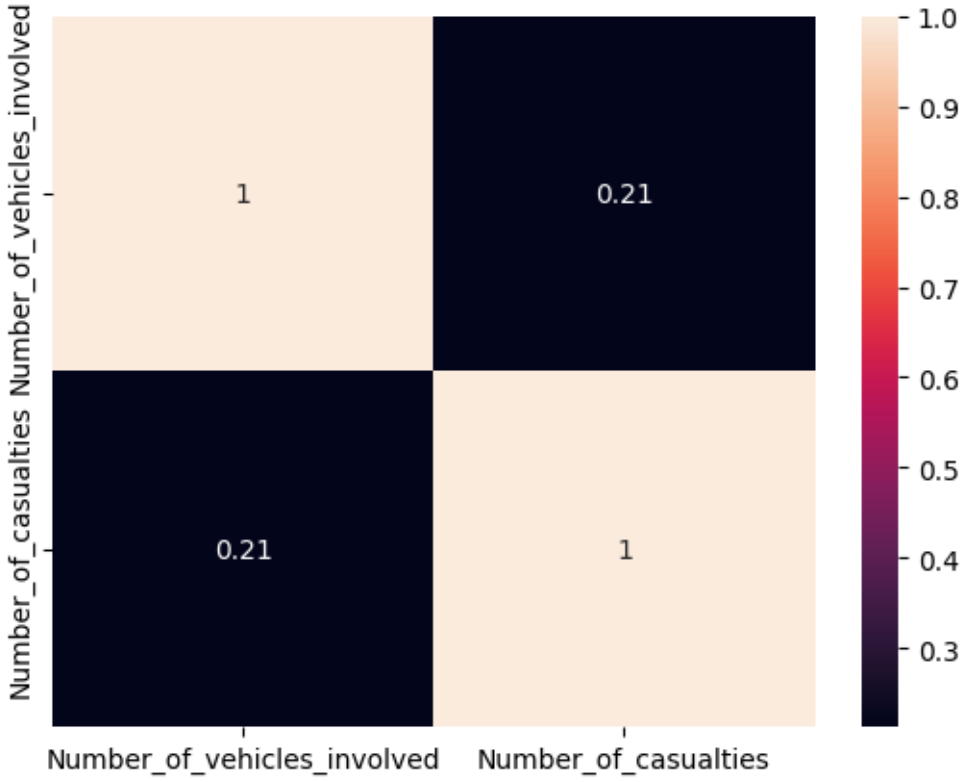
[ ]:
```
sns.scatterplot(x=df['Number_of_vehicles_involved'],␣
↪y=df['Number_of_casualties'])
plt.show()
```



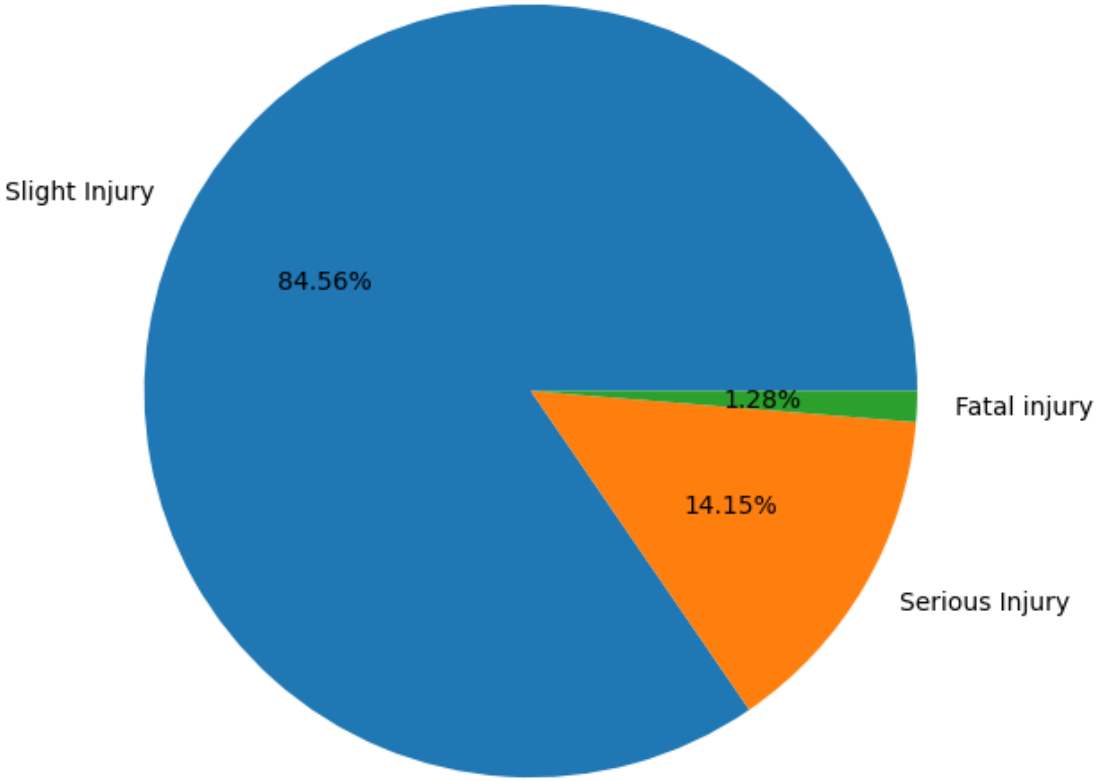
[ ]:
```
sns.pairplot(df[['Number_of_vehicles_involved','Number_of_casualties']])
plt.show()
```



[ ]:
```
correlation_matrix = df[['Number_of_vehicles_involved','Number_of_casualties']].
  ↪corr()
sns.heatmap(correlation_matrix, annot=True)
plt.show()
```



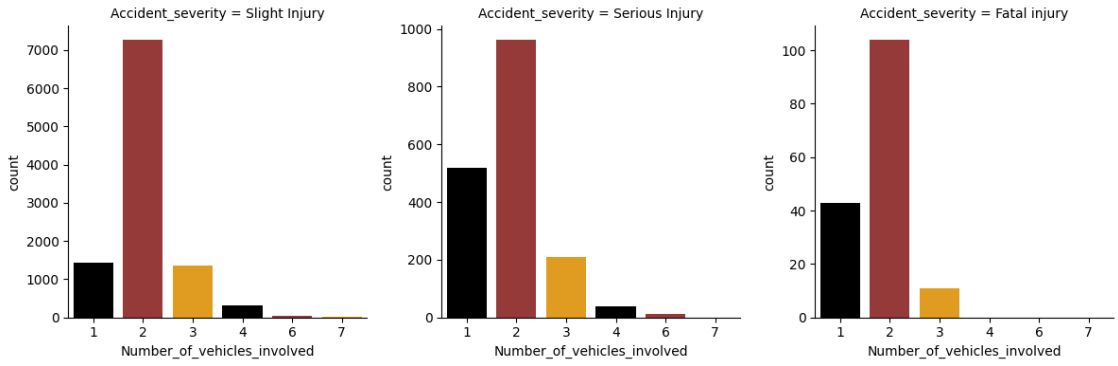
[ ]:
```
plt.figure(figsize=(10,7))
plt.pie(x=df['Accident_severity'].value_counts().values,
        labels=df['Accident_severity'].value_counts().index,
        autopct='%2.2f%%')
plt.show()
```



[ ]:
```
# creating a facet grid with columns as survived=0 and survived=1
grid = sns.FacetGrid(data=df, col='Accident_severity', height=4, aspect=1,␣
 ↪sharey=False)
# mapping bar plot and the data on to the grid
grid.map(sns.countplot, 'Number_of_vehicles_involved', palette=['black',␣
 ↪'brown', 'orange'])
plt.show()
```

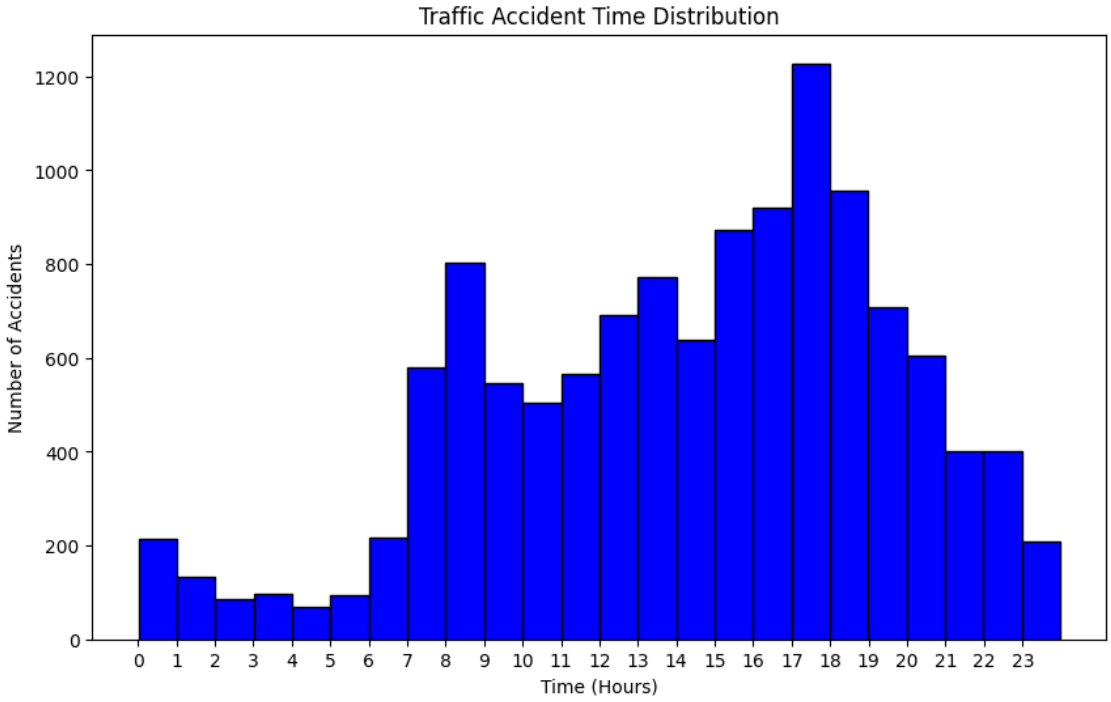
```
/usr/local/lib/python3.10/dist-packages/seaborn/axisgrid.py:718: UserWarning:
Using the countplot function without specifying `order` is likely to produce an
incorrect plot.
  warnings.warn(warning)
/usr/local/lib/python3.10/dist-packages/seaborn/axisgrid.py:854: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
```

```
effect.

  func(*plot_args, **plot_kwargs)
/usr/local/lib/python3.10/dist-packages/seaborn/axisgrid.py:854: UserWarning:
The palette list has fewer values (3) than needed (6) and will cycle, which may
produce an uninterpretable plot.
  func(*plot_args, **plot_kwargs)
/usr/local/lib/python3.10/dist-packages/seaborn/axisgrid.py:854: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  func(*plot_args, **plot_kwargs)
/usr/local/lib/python3.10/dist-packages/seaborn/axisgrid.py:854: UserWarning:
The palette list has fewer values (3) than needed (5) and will cycle, which may
produce an uninterpretable plot.
  func(*plot_args, **plot_kwargs)
/usr/local/lib/python3.10/dist-packages/seaborn/axisgrid.py:854: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  func(*plot_args, **plot_kwargs)
```



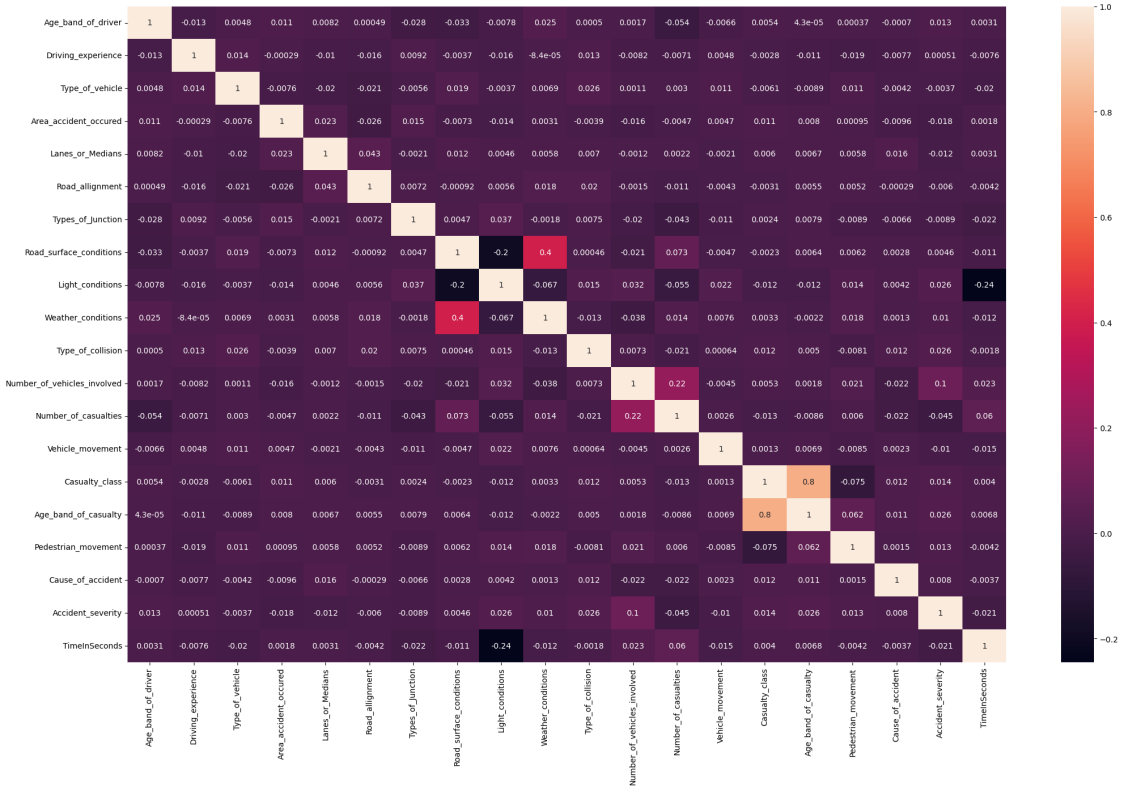
[ ]:
```
df['TimeInSeconds'] = df['Time'].apply(lambda x: x.hour * 3600 + x.minute * 60
 ↪+ x.second)

plt.figure(figsize=(10,6))
plt.hist(df['TimeInSeconds'], bins=24, color='blue', edgecolor='black')
plt.title('Traffic Accident Time Distribution')
plt.xlabel('Time (Hours)')
plt.ylabel('Number of Accidents')
```

```
plt.xticks(range(0, 24 * 3600, 3600), range(24))
plt.show()
```



[ ]:
```
plt.figure(figsize=[25,15])
sns.heatmap(df.corr(),annot=True)
```

[ ]: <Axes: >



FEATURE ENGINEERING

[ ]:
```
# dropping columns that can cause imbalance while imputation
lists=['Vehicle_driver_relation', 'Work_of_casuality',␣
  ↪'Fitness_of_casuality','Day_of_week','Casualty_severity','Time','Sex_of_driver','Educationa
  ↪'Road_surface_type','Sex_of_casualty']
df.drop(columns = lists, inplace=True)
```

[ ]:
```
df.columns
```

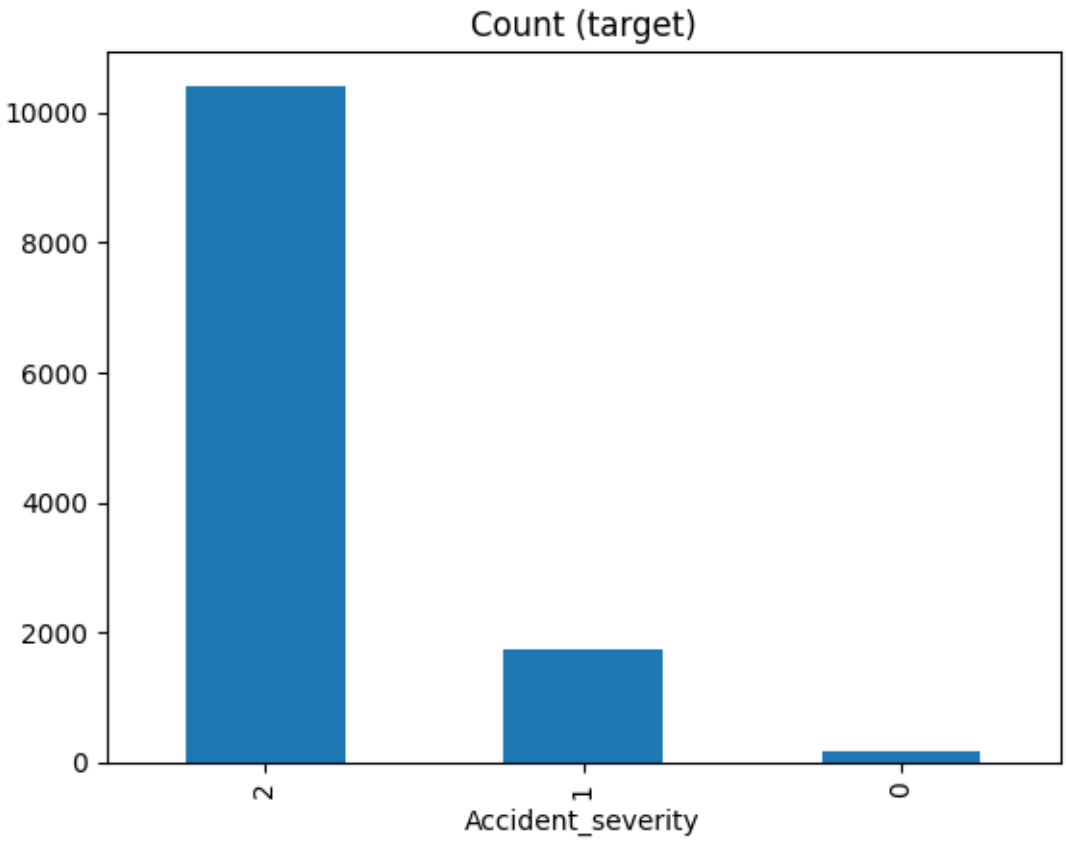
[ ]:
```
Index(['Age_band_of_driver', 'Driving_experience', 'Type_of_vehicle',
       'Area_accident_occured', 'Lanes_or_Medians', 'Road_allignment',
       'Types_of_Junction', 'Road_surface_conditions', 'Light_conditions',
       'Weather_conditions', 'Type_of_collision',
       'Number_of_vehicles_involved', 'Number_of_casualties',
       'Vehicle_movement', 'Casualty_class', 'Age_band_of_casualty',
       'Pedestrian_movement', 'Cause_of_accident', 'Accident_severity',
       'TimeInSeconds'],
      dtype='object')
```

[ ]:
```
from sklearn.preprocessing import LabelEncoder           #or one hot encoder
LE = LabelEncoder()
```

```
df=df.apply(LE.fit_transform)
```

[ ]:
```
target_count = df['Accident_severity'].value_counts()
print('Class 0:', target_count[0])
print('Class 1:', target_count[1])
print('Proportion:', round(target_count[0] / target_count[1], 2), ': 1')

target_count.plot(kind='bar', title='Count (target)');
```

```
Class 0: 158
Class 1: 1743
Proportion: 0.09 : 1
```



UPSAMPLING

[ ]:
```
from sklearn.model_selection import train_test_split
```

[ ]:
```
x = df.drop('Accident_severity', axis=1)
y = df['Accident_severity']
```

```
xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size=0.3,␣
 ↪random_state=42)
print(xtrain.shape, xtest.shape, ytrain.shape, ytest.shape)
```

```
(8621, 19) (3695, 19) (8621,) (3695,)
```

[ ]:
```
from collections import Counter
from imblearn.over_sampling import SMOTE
```

[ ]:
```
# upsampling using smote

counter = Counter(ytrain)

print("==============================")

for k,v in counter.items():
    per = 100*v/len(ytrain)
    print(f"Class= {k}, n={v} ({per:.2f}%)")

oversample = SMOTE()
xtrain, ytrain = oversample.fit_resample(xtrain, ytrain)

counter = Counter(ytrain)

print("==============================")

for k,v in counter.items():
    per = 100*v/len(ytrain)
    print(f"Class= {k}, n={v} ({per:.2f}%)")

print("==============================")

print("Upsampled data shape: ", xtrain.shape, ytrain.shape)
```

```
==============================
Class= 2, n=7324 (84.96%)
Class= 1, n=1191 (13.82%)
Class= 0, n=106 (1.23%)
==============================
Class= 2, n=7324 (33.33%)
Class= 1, n=7324 (33.33%)
Class= 0, n=7324 (33.33%)
==============================
Upsampled data shape:  (21972, 19) (21972,)
```

SPLITTING AND MODEL TRAINING

[ ]:
```
x=df.drop(columns=["Accident_severity"])
y=df["Accident_severity"]
```

[ ]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier,␣
 ↪GradientBoostingClassifier
```

[ ]:
```
# models={"LogisticRegression":LogisticRegression(),
#         "DecisionTreeClassifier":DecisionTreeClassifier(),
#         "SVM":SVC(),
#         "KNeighborsClassifier":KNeighborsClassifier(),
#         "GNB":GaussianNB(),
#        "RandomForestClassifier":RandomForestClassifier(),
#         "AdaBoostClassifier":AdaBoostClassifier(),
#         "GradientBoostingClassifier":GradientBoostingClassifier(),
#         }
from sklearn.multiclass import OneVsRestClassifier
models = {
    "LogisticRegression": OneVsRestClassifier(LogisticRegression(C=1.0,␣
 ↪solver='liblinear', penalty='l2')),
    "DecisionTreeClassifier":␣
 ↪OneVsRestClassifier(DecisionTreeClassifier(criterion='entropy', max_depth=5,␣
 ↪min_samples_split=5)),
    "KNeighborsClassifier":␣
 ↪OneVsRestClassifier(KNeighborsClassifier(n_neighbors=5, weights='distance',␣
 ↪p=2)),
    "RandomForestClassifier":␣
 ↪OneVsRestClassifier(RandomForestClassifier(n_estimators=100,␣
 ↪criterion='entropy', max_depth=8)),
    "AdaBoostClassifier":␣
 ↪OneVsRestClassifier(AdaBoostClassifier(n_estimators=200,learning_rate=0.1)),
    "GradientBoostingClassifier":␣
 ↪OneVsRestClassifier(GradientBoostingClassifier(n_estimators=100,␣
 ↪learning_rate=0.1, max_depth=5))
}
```

MODEL EVALUATION

[ ]:
```
from sklearn.pipeline import Pipeline
from sklearn.metrics import classification_report,accuracy_score
from sklearn.preprocessing import MinMaxScaler, StandardScaler
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve
```

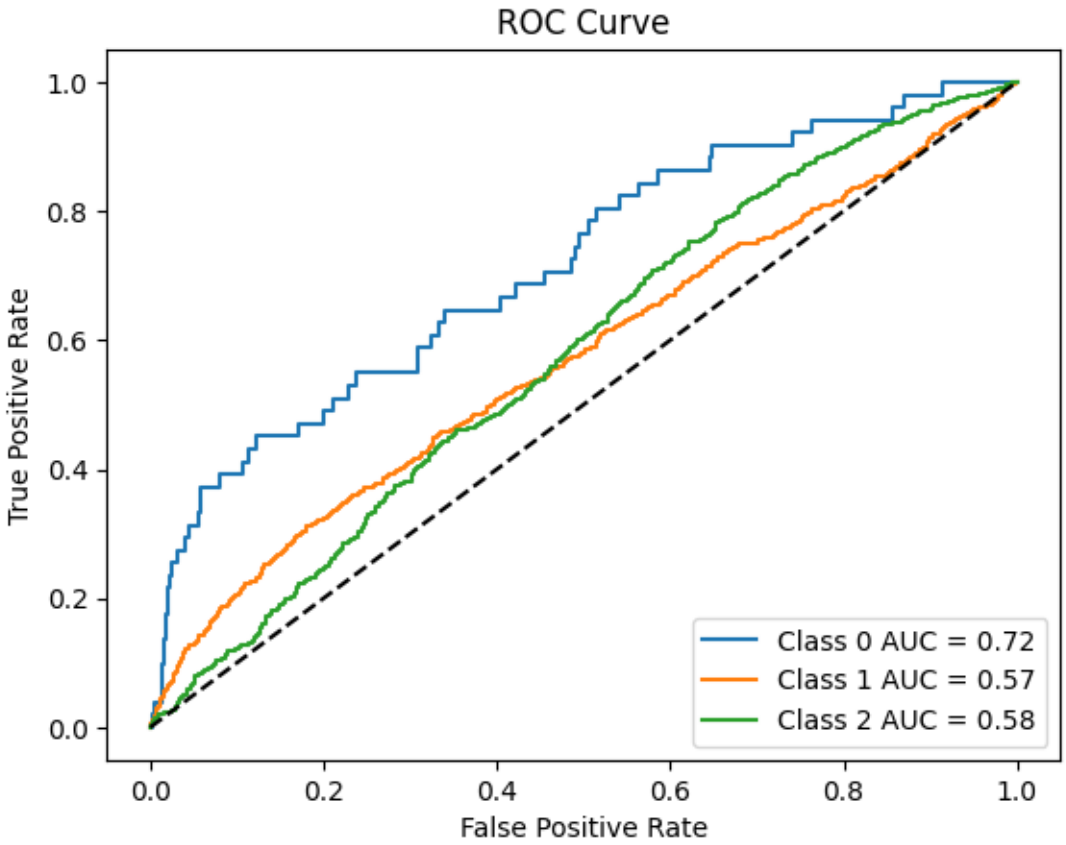
[ ]:
```
# models,x,y,scaleFlag=0,1,2
def modelAccuracy(models,x,y,scaleFlag):
    #train/Test
    xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.3,random_state=0)
    acc_result={}
    for name,model in models.items():
        #pipeline
        #1.Transformer -> 2.Model
        ␣
↪print("============================",name,"================================")
        if(scaleFlag==1):
            ␣
↪model_pipeline=Pipeline([('MinMaxScler',MinMaxScaler()),('model',model)])
        elif(scaleFlag==2):
            ␣
↪model_pipeline=Pipeline([('StandardScaler',StandardScaler()),('model',model)])
        else:
            model_pipeline=Pipeline([('model',model)])
        #training/testing on model pipeline
        model_fit=model_pipeline.fit(xtrain,ytrain)
        ypred=model_fit.predict(xtest)
        ypred_proba=model_fit.predict_proba(xtest)
        ypred_proba = np.nan_to_num(ypred_proba, nan=1/3)

        print(classification_report(ytest,ypred))
        auc =  roc_auc_score(ytest, ypred_proba, multi_class='ovr',␣
↪average='macro')
        print("macro-AUC:",auc)
        auc =  roc_auc_score(ytest, ypred_proba, multi_class='ovr',␣
↪average='weighted')
        print("weighted-AUC:",auc,"\n")
        plt_auc(ytest,ypred_proba)


acc=modelAccuracy(models,x,y,1)
```

```
============================ LogisticRegression ================================
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

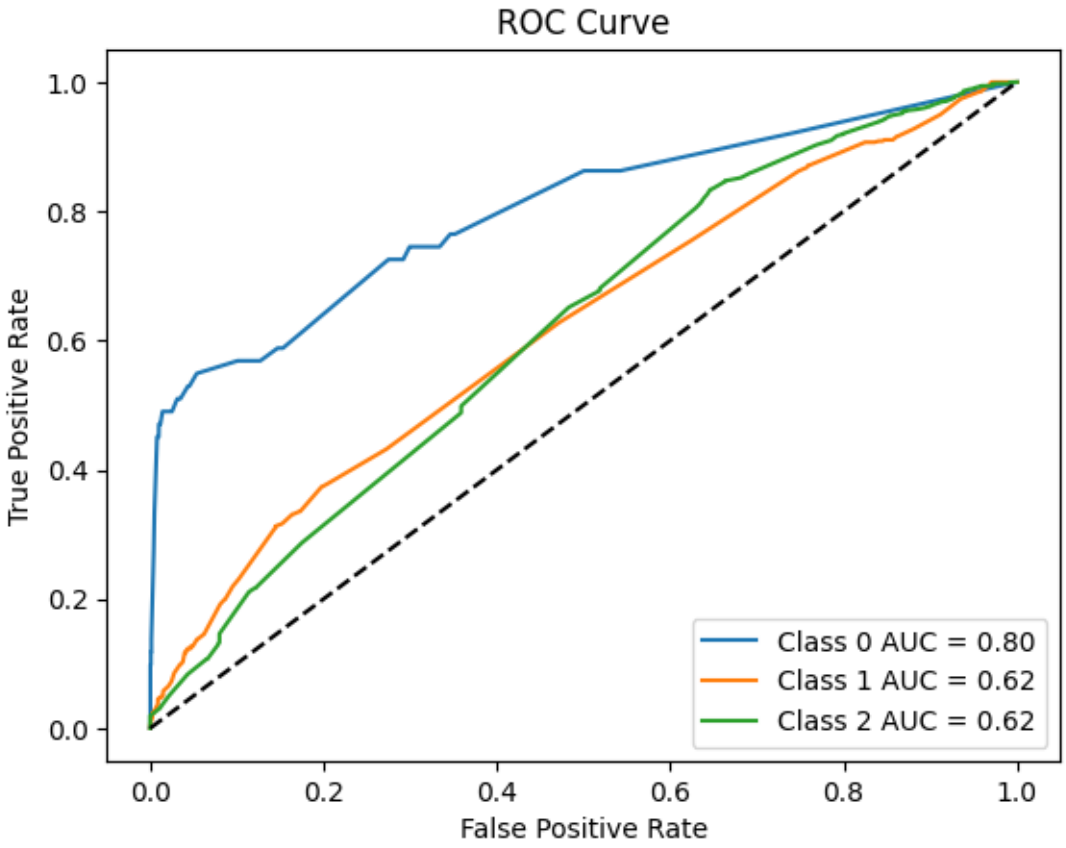
	precision	recall	f1-score	support
0	0.00	0.00	0.00	51
1	0.00	0.00	0.00	537
2	0.84	1.00	0.91	3107
accuracy			0.84	3695
macro avg	0.28	0.33	0.30	3695
weighted avg	0.71	0.84	0.77	3695

```
macro-AUC: 0.6213067642957238
weighted-AUC: 0.5778686446039931
```



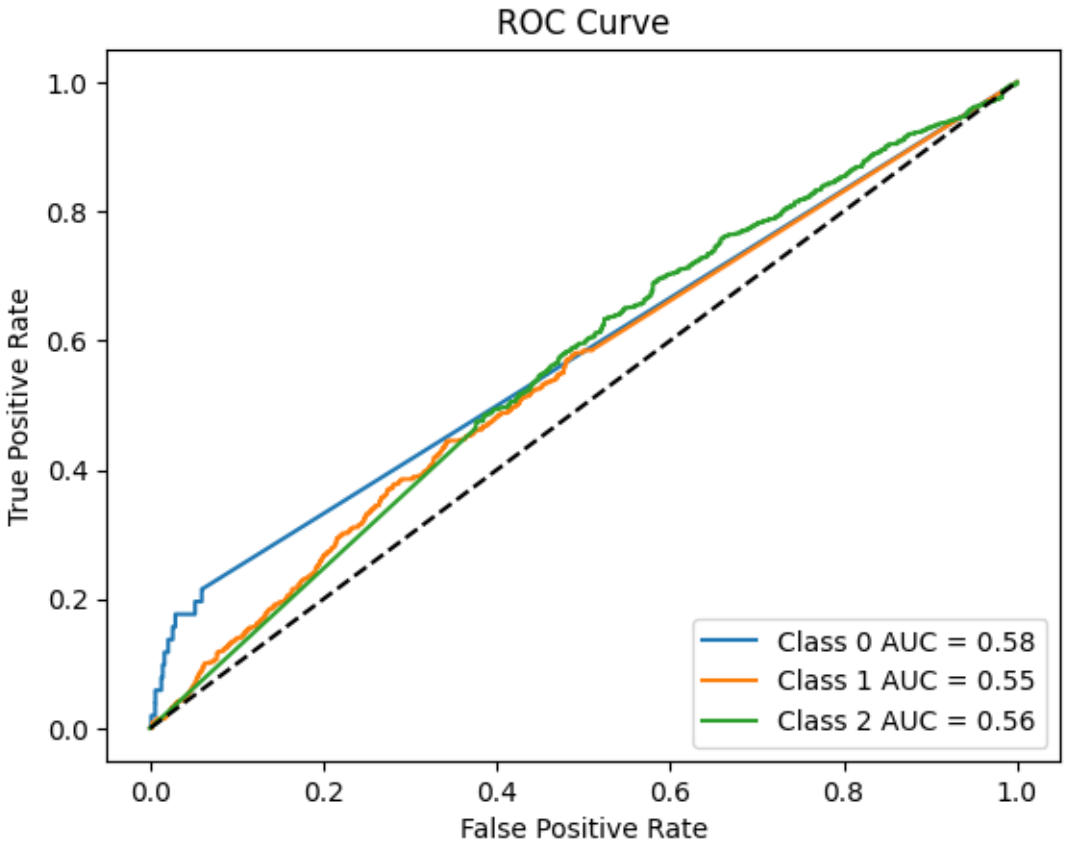

```
============================== DecisionTreeClassifier
```

```
================================
              precision    recall  f1-score   support

           0       0.83      0.10      0.18        51
           1       0.53      0.02      0.04       537
           2       0.84      1.00      0.91      3107

    accuracy                           0.84      3695
   macro avg       0.73      0.37      0.38      3695
weighted avg       0.80      0.84      0.78      3695

macro-AUC: 0.6792863278673856
weighted-AUC: 0.620070910304444
```




```
============================== KNeighborsClassifier
================================
              precision    recall  f1-score   support

           0       0.00      0.00      0.00        51
```

```
              1       0.15      0.02      0.04       537
              2       0.84      0.98      0.90      3107

    accuracy                            0.83      3695
   macro avg       0.33      0.33      0.31      3695
weighted avg       0.73      0.83      0.77      3695

macro-AUC: 0.5631938253966587
weighted-AUC: 0.558606823502534
```

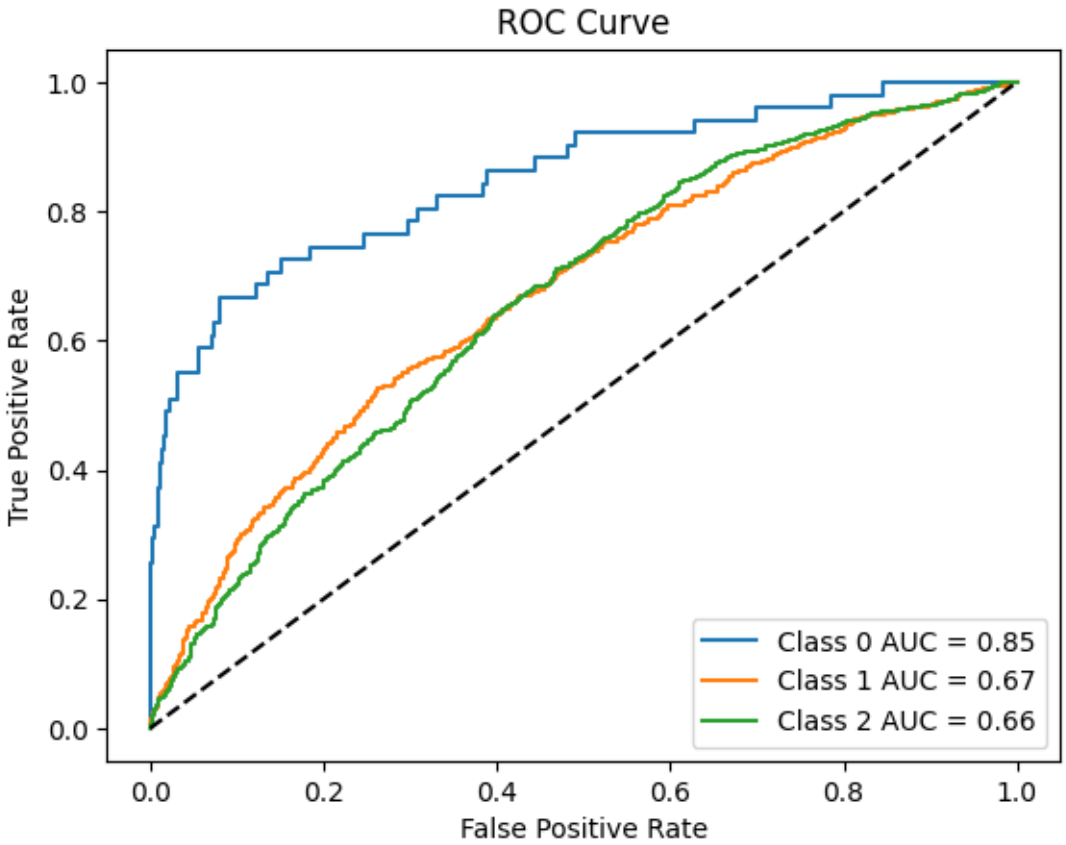



```
============================== RandomForestClassifier
================================

/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
```

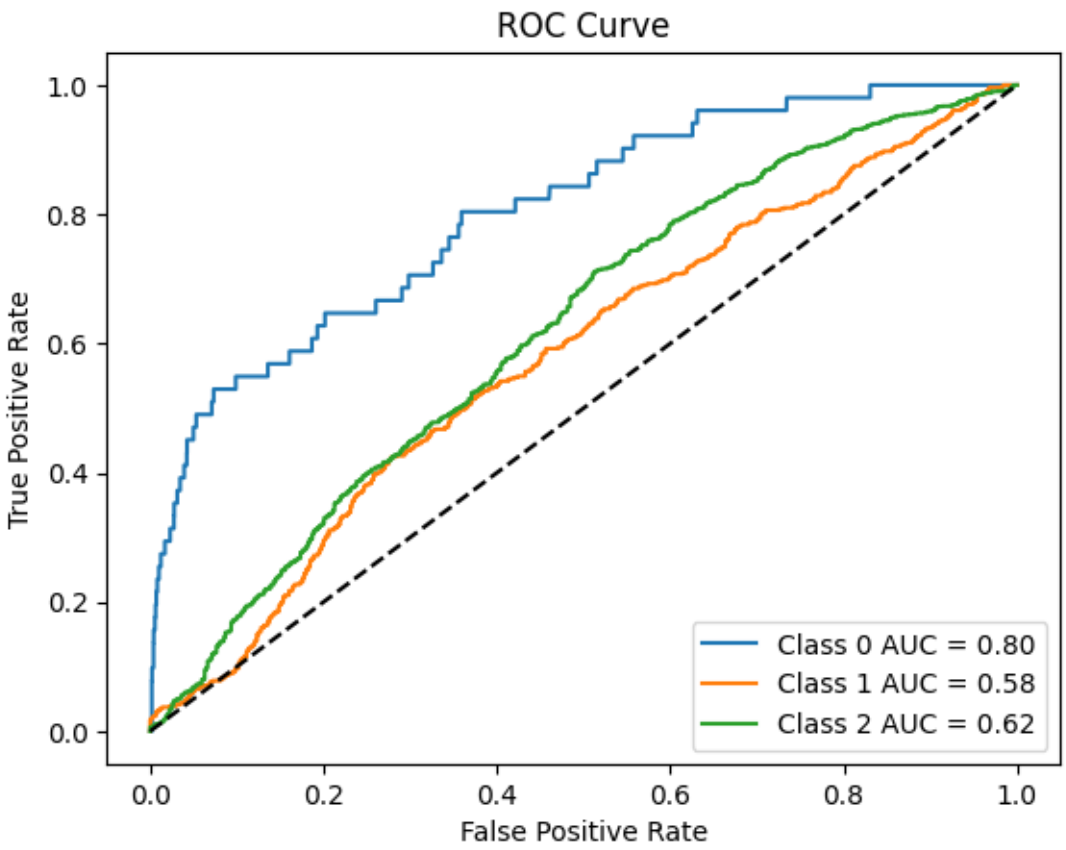
```
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

	precision	recall	f1-score	support
0	0.00	0.00	0.00	51
1	1.00	0.00	0.00	537
2	0.84	1.00	0.91	3107
accuracy			0.84	3695
macro avg	0.61	0.33	0.31	3695
weighted avg	0.85	0.84	0.77	3695

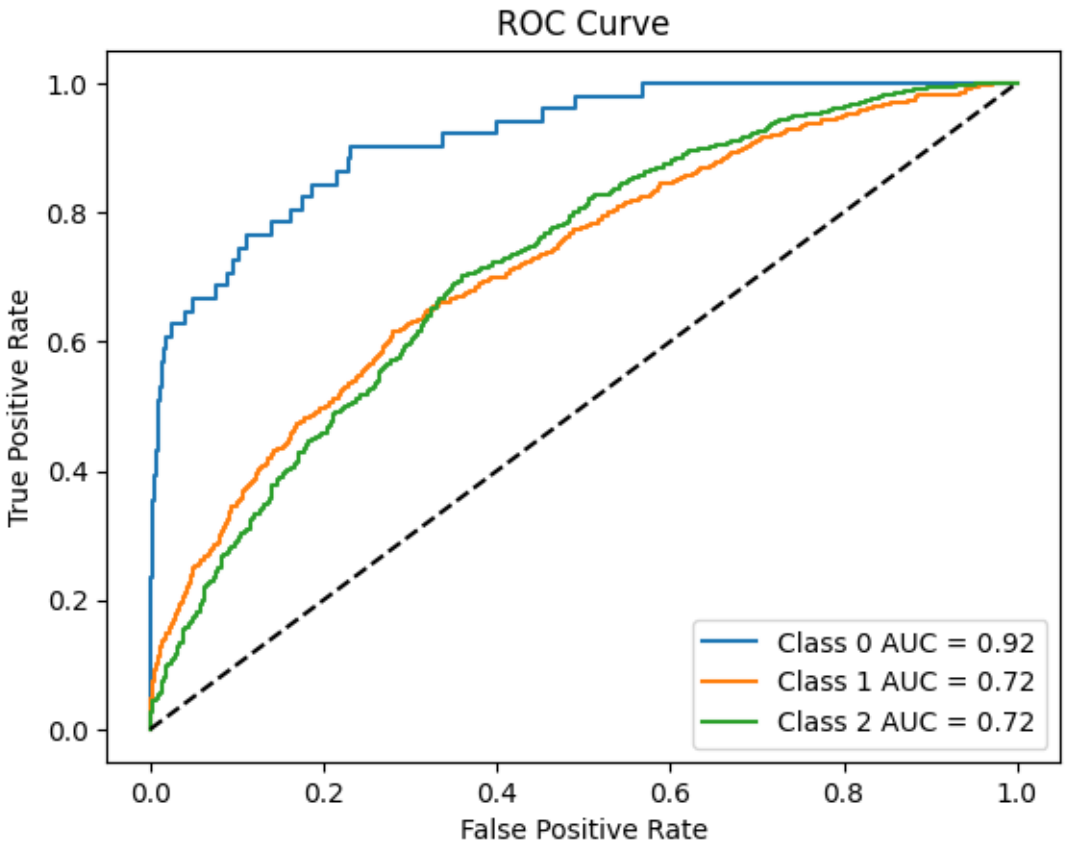
```
macro-AUC: 0.7285297276891907
weighted-AUC: 0.6661154702181663
```



```
============================ AdaBoostClassifier =================================
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
              precision    recall  f1-score   support

           0       0.00      0.00      0.00        51
           1       0.00      0.00      0.00       537
           2       0.84      1.00      0.91      3107

    accuracy                           0.84      3695
   macro avg       0.28      0.33      0.30      3695
weighted avg       0.71      0.84      0.77      3695

macro-AUC: 0.6680061303289714
weighted-AUC: 0.6169814922826339
```




```
============================== GradientBoostingClassifier
================================
              precision    recall  f1-score   support

           0       0.70      0.14      0.23        51
           1       0.72      0.09      0.16       537
           2       0.85      0.99      0.92      3107

    accuracy                           0.85      3695
   macro avg       0.76      0.41      0.44      3695
weighted avg       0.83      0.85      0.80      3695

macro-AUC: 0.7843804166417275
weighted-AUC: 0.7213491590919057
```


ROC Curve
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
Class 0 AUC = 0.92
Class 1 AUC = 0.72
Class 2 AUC = 0.72